```
import numpy as np
from sklearn.naive_bayes import GaussianNB
import pandas as pd
import matplotlib.pyplot as plt


df = pd.read_csv("/content/data_vowel_bayes.csv",names=['x1','x2','x3','T'])


df
```

	x1	x2	x3	T
0	700	1500	2600	1
1	550	1550	2400	1
2	700	1500	2600	1
3	700	1600	2700	1
4	550	1600	2600	1
...	...	...	...	...
866	500	1050	2900	6
867	500	1000	3000	6
868	500	1000	2800	6
869	500	900	2800	6
870	500	950	2700	6

871 rows × 4 columns

```
# we need to make x and y
y = df['T']
y
```

```
0      1
1      1
2      1
3      1
4      1
      ..
866    6
867    6
868    6
869    6
870    6
Name: T, Length: 871, dtype: int64
```

```
x = df.drop(['T'],axis=1)
x
```

	x1	x2	x3
0	700	1500	2600
1	550	1550	2400
2	700	1500	2600
3	700	1600	2700
4	550	1600	2600
...	...	...	...
866	500	1050	2900
867	500	1000	3000
868	500	1000	2800
869	500	900	2800
870	500	950	2700

871 rows × 3 columns

```
# we need to get the x value to be minmaxnormalisation
x = (x-x.min())/(x.max()-x.min())
x
```

	x1	x2	x3
0	0.692308	0.432432	0.571429
1	0.461538	0.459459	0.428571
2	0.692308	0.432432	0.571429
3	0.692308	0.486486	0.642857
4	0.461538	0.486486	0.571429
...	...	...	...
866	0.384615	0.189189	0.785714
867	0.384615	0.162162	0.857143
868	0.384615	0.162162	0.714286
869	0.384615	0.108108	0.714286
870	0.384615	0.135135	0.642857

871 rows × 3 columns

```
# splitting x and y into train and test set
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=20)


# fit it into the Gaussian model
model = GaussianNB()
model.fit(x_train,y_train)
```

```
GaussianNB()
```

```
# predict the value for y_pred
y_pred = model.predict(x_test)


# now we need to find the metrics of it
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
print(accuracy_score(y_test,y_pred))
```

```
0.7
```

```
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

```
                   precision    recall  f1-score   support

                1       0.50      0.50      0.50         2
                2       1.00      1.00      1.00         1
                3       1.00      0.50      0.67         2
                4       0.57      0.80      0.67         5
                5       0.67      1.00      0.80         4
                6       1.00      0.50      0.67         6

         accuracy                           0.70        20
        macro avg       0.79      0.72      0.72        20
     weighted avg       0.78      0.70      0.69        20

     [[1 0 0 0 1 0]
      [0 1 0 0 0 0]
      [0 0 1 0 1 0]
      [1 0 0 4 0 0]
      [0 0 0 0 4 0]
      [0 0 0 3 0 3]]
```

```
P{}
```